CS 164, Fall 2006 **CS 164: Homework #3** **P. N. Hilfinger**

Due: Monday, 2 October 2006

New instructions about homework. I'm switching to using a separate repository for homework. Check out homework framework with the command:

```
svn checkout svn+ssh://cs61b-tb@{\it HOST\/}/_hw/{\it LOGIN}
```

where *LOGIN* is your instructional login. Submit homework by committing the `hw3` subdirectory.

Unless the problem specifies otherwise, please put your solutions in a file named `hw3.txt` in your `hw3` subdirectory.

1. [From Aho, Sethi, Ullman] Indicate what language is described by each of the following grammars. In each case, S is the only non-terminal. Some symbols are quoted to make it clear that they are terminals.

a. S $\rightarrow$ 0 S 1 | 0 1

b. S $\rightarrow$ + S S | - S S | a

c. S $\rightarrow$ S "(" S ")" S | ϵ

d. S $\rightarrow$ a S b S | b S a S | ϵ

e. S $\rightarrow$ a | S + S | S S | S "*" | "(" S ")"

2. Identify each ambiguous grammar in problem 1 above, and give an unambiguous grammar that recognizes the same language (any such grammar—don't worry about associativity or precedence, since there are no semantic actions.)

3. For 1d above, give two distinct leftmost derivations for the string *abab*. For each derivation, show the corresponding parse tree and the rightmost derivation for that same parse tree.

4. [From Aho, Sethi, Ullman] Show that all binary (base 2) numerals produced by the following grammar denote numbers that are divisible by 3:

N $\rightarrow$ 11 | 1001 | N 0 | N N

Does this grammar generate all non-negative binary numerals that are divisible by 3?

5. A context-free grammar is *regular* if every production has either the form $A \rightarrow xB$, or the form $A \rightarrow x$, where A and B are non-terminals and x is a string of 0 or more terminal symbols. Show that the language described by any such grammar can be recognized by a NDFA (hence the term *regular*).

6. [From Aho, Sethi, Ullman] Try to design a context-free grammar for each of the following languages (it is not always possible). Whenever possible, make it a regular grammar.

a. The set of all strings of 0's and 1's where every 0 is immediately followed by at least one 1.

b. Strings of 0's and 1's with an equal number of 0's and 1's.

c. Strings of 0's and 1's with an unequal number of 0's and 1's.

d. Strings of 0's and 1's that do not contain the substring 011.

e. Strings of 0's and 1's of the form xy where x and y are equal-length strings and $x \neq y$.

f. Strings of 0's and 1's of the form xx.

7. Write a BNF grammar describing the language of boolean expressions whose value is `true`. The terminal symbols are '1' (true) , '0' (false), '`*`' (logical and), '`+`' (logical or), unary '`-`' (logical not) and left and right parentheses (for grouping). Assume the usual precedence rules, with logical "not" having highest precedence. That is, `1`, `1*1`, `1+0`, `1*1*-(0+1*0)`, and `-0` are all in the language, while `0`, `0+0`, prog—0*1—, and `1*1*(0+1*0)` are not. Your grammar may be ambiguous (that is, you may specify operator precedence and associativity separately). Put your solution in a file `7.y`. Start with the skeleton in `7.y` in the files for this homework assignment.

8. Consider the following ambiguous grammar:

`prog`	$\rightarrow$	ϵ
`prog`	$\rightarrow$	`expr ';'`
`expr`	$\rightarrow$	`ID`
`expr`	$\rightarrow$	`expr '-' expr`
`expr`	$\rightarrow$	`expr '/' expr`
`expr`	$\rightarrow$	`expr '?'  expr ':'  expr`
`expr`	$\rightarrow$	`'(' expr ')'`

The start symbol is `prog`; `ID` and the quoted characters are the terminals.

a. Produce an (improper) LL(1) parsing table for this grammar. Since it is ambiguous, some slots will have more than one production; list all of them. Show the FIRST and FOLLOW sets.

b. Modify the grammar to be LL(1) and repeat part a with it.

In this case, we're just interested in recognizing the language, so don't worry about preserving precedence and associativity.